

**Introduction**

**Preface:**“Our aim is to provide a Practical, Hands free, Safe, Unique, and 100% Independent Solution for the Visually Impaired”

**Information author:**Noureddine El Elani  
noor.elelani@student.thomasmore.be  
r0708198

**Appraisal to all contributors:**Mr. Marc Roggemans

**Intro:**

Not all people are born equal in their body abilities… With a boost of advanced technology, a help from brilliant programmers, social active helpers, social media, and advertisements, we will be able to assist people with weak abilities, at the moment we’ll be focusing on the visually impaired.

After all, according to the World Health Organization the number of the visually impaired people around the globe is more than 285 million, this big number cannot be reduced by a simple medical prescription, or a magical pill.

It is about time to make their dream a reality, allowing them to live as close as possible, to what we call a standard life.

And that’s why we are building both: a hardware, and a software program, aiming to help the people with little, or no eye sight, in performing their daily activities, freely, safely, 100% hands-free and with no help at all, whether it was a simple activity, such as going out for a cup of coffee with friends, or for even doing their daily full time job, We call it Pathfinder.

**Problems and targets:**Some of the problems that I encountered during the semester is trying to work around assembly language, trying to make cheap inaccurate sensors better and more accurate which is impossible the only thing I was able to do is to minimize the error, the problem that I faced the most was receiving defect components or frying components because of vision problems (lesson learnt, triple check before powering the device because my eyes could fool me).

How to detect stairs up/down.

I couldn’t 3d print a module because the school was closed.

**Strategy applied for tackling the problem:**By doing a lot of research and staying patient and persistent.

The problem of detecting stairs up or down, I used the two sensors, both sensors have to measure a certain distance to know whether it’s a stairs up or down.

The 3d printed problem was solved by making the module out of a cardboard, not perfect but better than not handing anything in…

**Content**

**Project proposal:**

Pathfinder is a fully automated visual aid and navigation system, running in parallel, via sensor-fitted belt in parallel with a smart phone application to provide both unique, and 100% independent experience to the visually impaired, including other features.

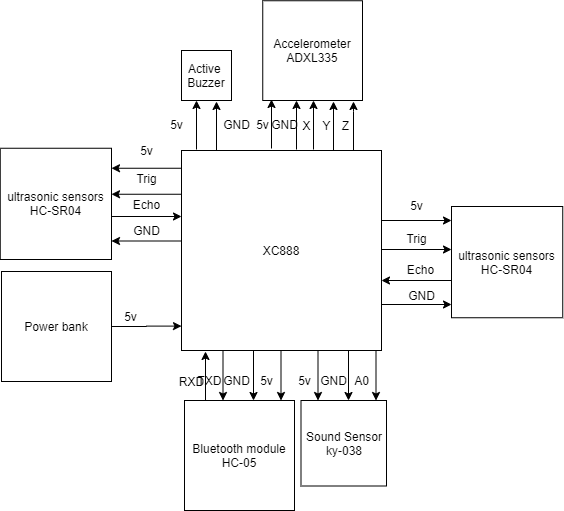
Some of the features that will be included in the device is a find/lost system, if the user lost his device somewhere in the room, all they have to do is just clap and the device will start beeping tills it’s moved.

Pathfinder will be the eyes and nerves of the visually impaired, feeding their ears and thus their brain, about their surroundings, to help them deal easily, freely, and safely with what is going on around them.

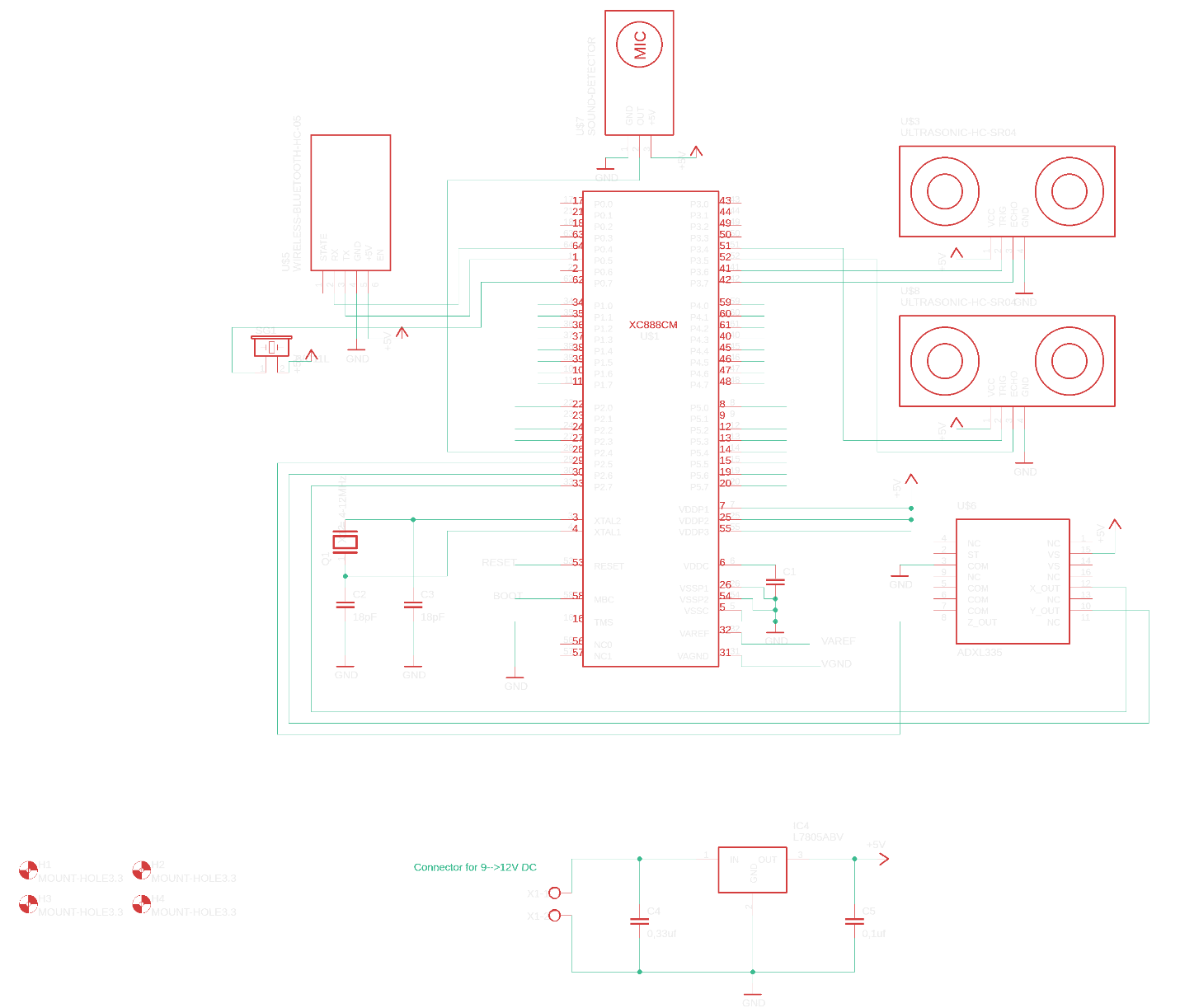
The device uses 2 ultrasonic sensor (HC-SR04) to detect obstacles such as chairs, tables, people, stairs up, stairs down etc.., the device will be connected to the smartphone application via a Bluetooth module (HC-05), the sound is being detected using a sound sensor (ky-038), an active buzzer is being used for the buzzing sound and an accelerometer (ADXL335) to detect movement, I’m using the Infineon XC888 as the microcontroller of choice.

**Hardware Documentation**

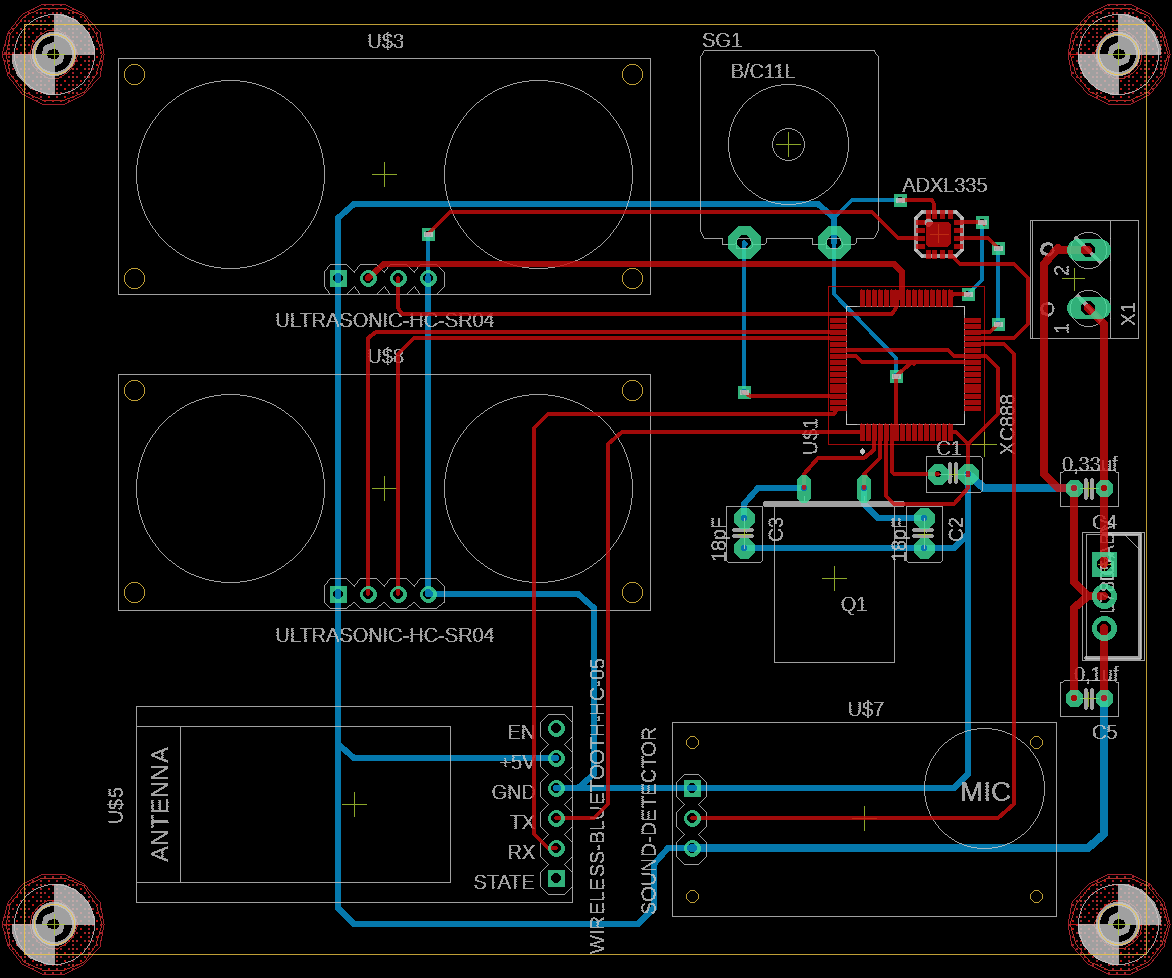
**Block schematic diagram:**



**Schematic diagram:**



**PCB layout:**

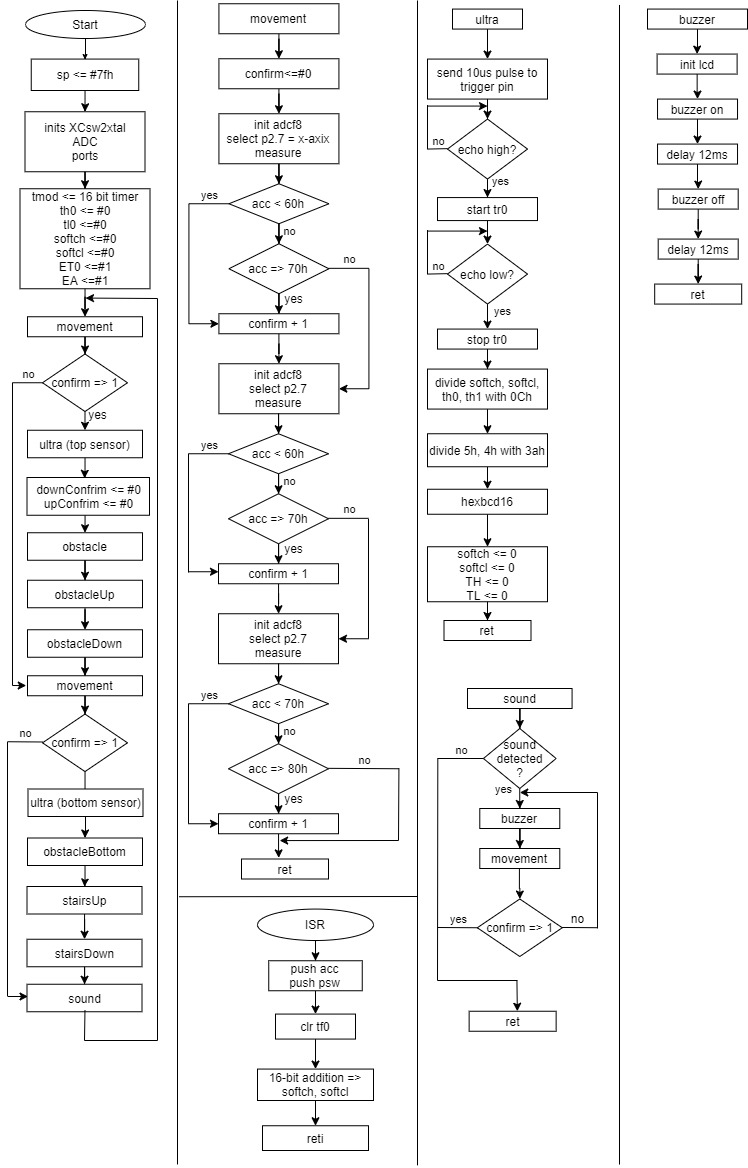


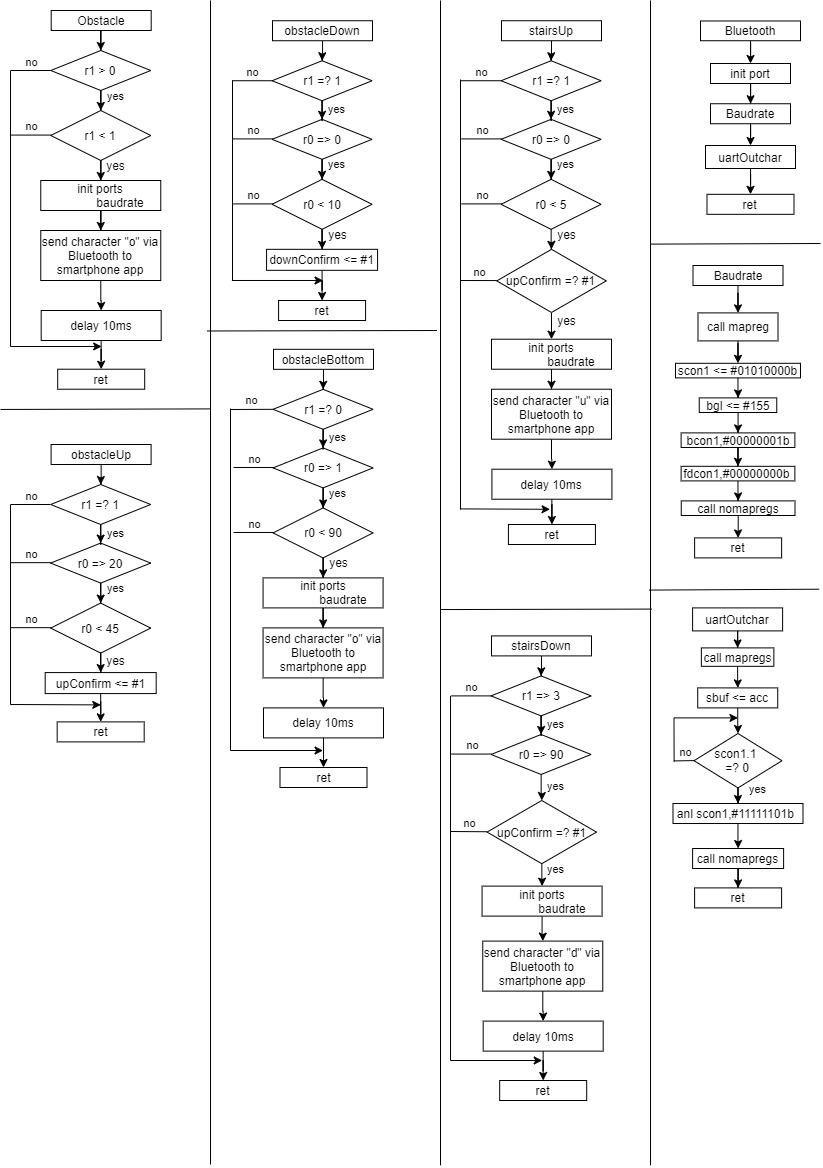
**Parts list with pricings (shipping included):**

* Infineon XC888 microcontroller: 70€
* Ultra Sonic sensors – HC-SR04 x2: 3 x 2 = 6€
  + <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
* Sound sensor – ky-038: 6€
  + <https://datasheetspdf.com/pdf-file/1402048/Joy-IT/KY-038/1>
* Bluetooth Module HC-05: 6€
  + <https://www.electronicaestudio.com/docs/istd016A.pdf>
* Accelorometer – ADXL335: 6€
  + <https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf>
* PCB: 14€
* Active buzzer: 2€
  + <http://www.electronicoscaldas.com/datasheet/LTE12-Series.pdf>
* Power Bank: 30€

**Software documentation**

**Flowchart:**





**Listing (source code in ASM):**

In the Rapport\_PE\_tussentijdse.pdf you left me this comment “In your code you state you'll get interrupts every microsecond. This implies your ISR must be executed in under 1 microsecond. The fastest instruction (1 instruction) takes 2x42ns=0.082us, the longest will take 8x42ns=336ns=0,33 microseconds. Your IRS (without you knowing) takes 0.336 microseconds to enter and 0,252 microseconds leave. So without any instructions executed the ISR takes 0,588microseconds!!In the code you test for the echo pin in a loop with a delay of 10 microseconds. This means that you can have a measurement error of 10 microseconds. This gives an error on your measurement of 17 cm.” when I measure using the ultrasonic sensor and I print the results to the LCD screen I get accurate results, I compared some measurements done by the sensor with manual measurements I did using a meter I got the exact same results.

Thank you for the feedback, and sorry about the messy code before...

**softch equ 10h**

**softcl equ 11h**

**confirm equ 12h**

**upConfirm equ 13h**

**downConfirm equ 14h**

**trig0 bit p3\_data.6**

**echo0 bit p3\_data.7**

**trig1 bit p3\_data.4**

**echo1 bit p3\_data.5**

**inSoundx bit p3\_data.2**

**org 0000h ;origin code in memory**

**ljmp start ;jump over memory locations containing interrupt vectors**

**org 000bh ;Every interrupt source(s) has a unique end specific vector**

**ljmp introutxyz ;memory space between vectors is limited. We use the vector**

**;location to store a ljmp instruction to the actual routine**

**start: mov sp,#7fh ;stackpointer initialisation**

**lcall inits ;inits**

**lcall echoinput ;set p3\_data inputs and outputs**

**lcall startprog ;timer and ISR inits**

**loop: lcall topSensorx ;top sensor**

**lcall bottomSensorx ;bottom sensor**

**lcall soundSensor ;sound sensor**

**ljmp loop**

**inits:**

**lcall XCsw2xtal ;swtich to crystal oscillator because it's more accurate**

**lcall initadc ;init**

**ret**

**echoinput:**

**push syscon0 ;select map**

**mov syscon0,#004h ;addressing the 7 bits**

**push port\_page ;save**

**mov port\_page,#001h ;select port page 1**

**mov p3\_pudsel,#05Bh ;select pull\_up device ;01011011**

**mov p3\_puden,#05Bh ;select on**

**mov port\_page,#000h ;select page 0**

**mov p3\_dir,#05Bh ;port 4 as input**

**pop port\_page ;retrieve**

**pop syscon0 ;retrieve**

**ret**

**startprog:**

**;initialize timer**

**mov tmod,#00000001b ;16-bit timer**

**mov TH0,#0 ;set timer high to 0**

**mov TL0,#0 ;set timer low to 0**

**mov softch,#00h ;set softch to 0**

**mov softcl,#00h ;set softcl to 0**

**setb ET0 ;interupt0 = 1**

**setb EA ;enable**

**clr TF0 ;clear timer flag 0**

**ret**

**topSensorx:**

**;acceloromter to check if there is movement detected**

**lcall movement ;call accelorometer to check if movemnet is detected**

**cjne a,#1,checkConfirm3 ;compare a = confirm to 1**

**checkConfirm3:**

**jnc movDet ;if a is = to 1 or bigger that means movement is detected so activate sensor otherwise return**

**ret**

**movDet:**

**;top sensor**

**jb echo0,skipDelay ;if echo pin is high skip delay else put trig pin high for 10us**

**lcall delay10us ;for 10us**

**setb trig0 ;put trig pin high, (works reversed)**

**lcall delay10us ;for 10us**

**clr trig0 ;then low**

**skipDelay:**

**jnb echo0,skipDelay ;if echo pin is low keep looping else run tr0, wait till echo pin is high**

**setb tr0 ;start timer 0**

**repeat:**

**;A timeout is not needed because in case the singal didn't return the sensor automatically puts the echo pin low so it will never have an inifinte loop.**

**jb echo0,repeat ;if echo pin is high keep looping else stop tr0.**

**clr tr0 ;stop timer 0**

**mov downConfirm,#0 ;move 0 to downConfrim**

**mov upConfirm,#0 ;move 0 to upConfrim**

**lcall calc**

**lcall checkObst ;osbstacle in front of the user detection using top sesnor**

**lcall checkobstup ;use top sensor to check step 1 for stairs up**

**lcall checkobstDown ;use top sensor to check step 1 for stairs down**

**ret**

**bottomSensorx:**

**lcall movement ;call accelorometer to check if movemnet is detected**

**cjne a,#1,checkConfirm2 ;compare a = confirm to 1**

**checkConfirm2:**

**jnc movDet2 ;if a is = to 1 or bigger that means movement is detected so activate sensor otherwise return**

**ret**

**movDet2:**

**;bottom sensor**

**jb echo1,skipDelay1 ;if echo pin is high skip delay else put trig pin high for 10us**

**lcall delay10us ;for 10us**

**setb trig1 ;put trig pin high, (works reversed)**

**lcall delay10us ;for 10us**

**clr trig1 ;then low**

**skipDelay1:**

**;A timeout is not needed because in case the singal didn't return the sensor automatically puts the echo pin low so it will never have an inifinte loop.**

**jnb echo1,skipDelay1 ;if echo pin is low keep looping else run tr0, wait till echo pin is high**

**setb tr0 ;start timer 0**

**repeat1:**

**jb echo1,repeat1 ;if echo pin is high keep looping else stop tr0.**

**clr tr0 ;stop timer 0**

**lcall calc**

**lcall obstbot ;osbstacle in front of the user detection using bottom sesnor**

**lcall stairsUp ;use bottom sensor to check step 2 for stairs up**

**lcall stairsDown ;use bottom sensor to check step 2 for stairs down**

**ret**

**;soundSesnor**

**soundSensor:**

**jnb inSoundx,loopBack ;if no sound detected don't activate buzzer otherwise do**

**;buzzer**

**onSound:**

**lcall buzzerx ;to activate buzzer**

**;acceloromter**

**accelorometer:**

**;lcall movement ;call accelorometer to check if movemnet is detected**

**cjne a,#1,checkConfirm ;compare a = confirm to 1**

**checkConfirm:**

**jnc loopBack ;if no movement is detected keep buzzer on otherwise turn off**

**onSoundBack:**

**ljmp onSound**

**loopBack:**

**ret**

**;accelorometer**

**movement:**

**mov confirm,#0 ;set confirm to 0**

**lcall accX ;X-axis**

**lcall accY ;Y-axis**

**lcall accZ ;Z-axis**

**mov a,confirm ;set a = to confirm**

**ret**

**calc:**

**mov r3,softch ;store mesaured time in registers**

**mov r2,softcl ;idem**

**mov r1,th0 ;idem**

**mov r0,tl0 ;idem**

**mov r7,#00h**

**mov r6,#00h**

**mov r5,#00h**

**mov r4,#0Ch ;12**

**lcall div32 ;divide, result in r4 (time in microseconds and lower than 6553, r3,r2,r1,r0 / r7,r6,r5,r4 Q=r7,r6,r5,r4 R=r3,r2,r1,r0**

**;r4 contain the timer value in us**

**mov r1,05h ;timer in us**

**mov r0,04h**

**mov r3,#00h**

**mov r2,#3ah ;58 = speed of sound in (cm/us)/2 (divided by two because the wave has to go out and come back so we cut in half)**

**lcall div16 ;time in us/58**

**mov r1,03h ;result (meter result)**

**mov r0,02h ;result (cm result)**

**lcall hexbcd16 ;converts 16 bit hex number to BCD notation**

**mov softch,#00h ;set softch to 0**

**mov softcl,#00h ;set softcl to 0**

**mov TH0,#0 ;set timer high to 0**

**mov TL0,#0 ;set timer low to 0**

**ret**

**;osbstacle in front of the user detection using top sesnor**

**checkObst:**

**cjne r1,#0,checkObstBefore**

**checkObstBefore:**

**jc noObst ;if r1 is less than 0 (0 meters) check second condition else skip,**

**cjne r1,#1,checkObstafter ;I know it seems stupid to check below 0 but this is the only way it worked**

**checkObstafter:**

**jnc noObst ;if r1 is less than 0 (0 meters) check second condition else skip**

**lcall inituart1 ;sio and ports, (I had to call uart under and not initialize it above because I'm using the buzzer so they effect each other)**

**mov a,#6Fh ;send character o = obstacle**

**lcall uart1outchar ;routine to send character**

**mov a,#10**

**lcall delaya0k05s ;delay between each character to avoid colision**

**noObst:**

**ret**

**;use top sensor to check step 1 for stairs up**

**checkobstup:**

**cjne r1,#1,noobstUp ;if r1 is equal to 1 (1 meter) continue otherwise skip**

**cjne r0,#20,checkUpobstbefore**

**checkUpobstbefore:**

**jc noobstUp ;if r0 is equal to or greater than 20 (.20 meters) check second condition else skip**

**cjne r0,#45,checkUpobstafter**

**checkUpobstafter:**

**jnc noobstUp ;if r0 is less than 44 (.44 meters) move 1 to upConfirm else skip**

**mov upConfirm,#1 ;mov 1 to upConfirm, used to make sure**

**noobstUp:**

**ret**

**;use top sensor to check step 1 for stairs down**

**checkobstdown:**

**cjne r1,#1,noobstDown ;if r1 is equal to 1 (1 meter) continue otherwise skip**

**cjne r0,#0,checkObstDownBefore**

**checkObstDownBefore:**

**jc noobstDown ;if r0 is equal to or greater than 0 (0 meters) check second condition else skip**

**cjne r0,#10,checkObstDownAfter**

**checkObstDownAfter:**

**jnc noobstDown ;if r0 is less than 10 (.10 meters) check second condition else skip**

**mov downConfirm,#1 ;mov 1 to downConfirm, used to make sure**

**noobstDown:**

**ret**

**;osbstacle in front of the user detection using bottom sesnor**

**obstbot:**

**cjne r1,#0,noobstbot ;if r1 is not equal to 0 (0 meters) skip else continue**

**cjne r0,#1,checkObstBotBefore**

**checkObstBotBefore:**

**jc noobstbot ;if r0 is equal to or greater than 1 (.01 meters) check second condition else skip**

**cjne r0,#90,checkObstBotAfter**

**checkObstBotAfter:**

**jnc noObstBot ;if r0 is less than 90 (.90 meters) send character else skip**

**lcall inituart1 ;sio and ports**

**mov a,#6fh ;send character u = stairs up**

**lcall uart1outchar ;routine to send character**

**mov a,#10**

**lcall delaya0k05s ;delay between each character to avoid colision**

**noObstBot:**

**ret**

**;use bottom sensor to check step 2 for stairs up**

**stairsUp:**

**cjne r1,#1,noUp ;if r1 is not equal to 1 (1 meters) skip else continue**

**cjne r0,#0,checkUpAfter**

**checkUpAfter:**

**jc noUp ;if r0 is less than 0 (.0 meters) check second condition else skip**

**cjne r0,#5,checkUpAfter0**

**checkUpAfter0:**

**jnc noUp ;if r0 is less than 5 (.05 meters) send character else skip**

**mov a,upConfirm ;move upConfirm to a**

**cjne a,#1,noUp ;if upConfirm = 1 send character else skip**

**lcall inituart1 ;sio and ports**

**mov a,#75h ;send character u = stairs up**

**lcall uart1outchar ;routine to send character**

**mov a,#10**

**lcall delaya0k05s ;delay between each character to avoid colision**

**noUp:**

**ret**

**;use bottom sensor to check step 2 for stairs down**

**stairsDown:**

**cjne r1,#3,checkDownBefore0**

**checkDownBefore0:**

**jc noDown ;if r1 is equal to or greater than 3 (3 meters) check second condition else skip**

**cjne r0,#90,checkDownBefore**

**checkDownBefore:**

**jc noDown ;if r0 is equal to or greater than 90 (.90 meter) check downConfirm else skip**

**mov a,downConfirm ;move downConfirm to a**

**cjne a,#1,noDown ;if downConfirm = 1 send character else skip**

**lcall inituart1 ;sio and ports**

**mov a,#64h ;send character d = stairs down**

**lcall uart1outchar ;routine to send character**

**mov a,#10**

**lcall delaya0k05s ;delay between each character to avoid colision**

**noDown:**

**ret**

**;x-axis of acceloromoter**

**accX: lcall initadcf8 ;initialise ADC for basic usage (only getadc4to7 can be used) overclocked 10 bit resolution**

**mov a,#10000000b ;select p2.7**

**lcall getadc4to7 ;measure**

**cjne a,#60h,checkXBefore**

**checkXBefore:**

**jc noX ;if a is less than 60h add 1 to confirm else check second condition**

**cjne a,#70h,checkXAfter**

**checkXAfter:**

**jc skipX ;if a is equal to or greater than 70h add 1 to confirm else skip**

**noX: mov a,#1**

**mov b,confirm**

**add a,b**

**mov confirm,a**

**skipX:**

**ret**

**;y-axis of accelorometer**

**accY: lcall initadcf8 ;initialise ADC for basic usage (only getadc4to7 can be used) overclocked 10 bit resolution**

**mov a,#01000000b ;select p2.6**

**lcall getadc4to7 ;measure**

**cjne a,#60h,checkYBefore**

**checkYBefore:**

**jc noY ;if a is less than 60h add 1 to confirm else check second condition**

**cjne a,#70h,checkYAfter**

**checkYAfter:**

**jc skipY ;if a is equal to or greater than 70h add 1 to confirm else skip**

**noY: mov a,#1**

**mov b,confirm**

**add a,b**

**mov confirm,a**

**skipY:**

**ret**

**;z-axis of accelorometer**

**accZ: lcall initadcf8 ;initialise ADC for basic usage (only getadc4to7 can be used) overclocked 10 bit resolution**

**mov a,#00100000b ;select p2.5**

**lcall getadc4to7 ;measure**

**cjne a,#70h,checkZBefore**

**checkZBefore:**

**jc noZ ;if a is less than 70h add 1 to confirm else check second condition**

**cjne a,#80h,checkZAfter**

**checkZAfter:**

**jc skipZ ;if a is equal to or greater than 80h add 1 to confirm else skip**

**noZ: mov a,#1**

**mov b,confirm**

**add a,b**

**mov confirm,a**

**skipZ:**

**ret**

**;buzzer**

**buzzerx:**

**lcall initlcd ;initailize lcd for buzzer, lcd must be initialized, because uart will interupt the buzzer port**

**lcall lcdbuzon ;buzzer on**

**mov a,#12**

**lcall delaya0k05s ;delay**

**lcall lcdbuzoff ;off**

**mov a,#12**

**lcall delaya0k05s ;delay**

**ret**

**inituart1:**

**lcall ports ;set port**

**lcall baudrate ;set baud rate**

**ret**

**; port p0.5 as input**

**; poort p0.4 as output**

**ports: push port\_page ;backup port\_page state**

**mov port\_page,#00h ;select page 0**

**mov p0\_dir,#00010000b ;p0.4 = output**

**mov p0\_data,#ffh ;all outputs on one**

**mov port\_page,#02 ;select page 2**

**mov p0\_altsel0,#00010000b ;altsel for p0.4**

**mov p0\_altsel1,#00010000b ;idem**

**pop port\_page ;retrieve port\_page old state**

**ret**

**baudrate:**

**lcall mapregs**

**mov scon1,#01010000b ;initialize UART**

**;first initialize BG, then bcon, otherwise BG won't be used**

**mov bg1,#155 ;don't use fraction divider**

**mov bcon1,#00000001b ;ativate baud rate generator**

**mov fdcon1,#00000000b ;fractional divider is off**

**lcall nomapregs**

**ret**

**uart1outchar:**

**lcall mapregs ;regs of uart1 sit in the mapped zone**

**MOV SBUF1,A ;send character**

**uart1outchar1:**

**JNB scon1.1,uart1outchar1 ;wait till sender is available**

**;CLR scon1.1 ;clear bit**

**;\*\*\*\*\* clr scon1.x instruction because of a bug in the chip!!**

**;\*\*\*\*\* we solve it using anl scon1,#11111101b**

**anl scon1,#11111101b ;alternative for clear TI flag**

**lcall nomapregs ;back to standard set regs**

**ret**

**introutxyz: ;subroutine**

**;no infinite loops in interupts**

**push acc**

**push psw**

**;everytime there's an overflow an interupt occurs thus inc 1**

**;16 bit addition**

**clr tf0 ;clear timer flag 0**

**mov a,softcl ;mov softcl to a**

**add a,#1 ;add 1 to a**

**mov softcl,a ;write low result back**

**mov a,softch ;high byte addition, use carry**

**addc a,#0**

**mov softch,a ;write high result back**

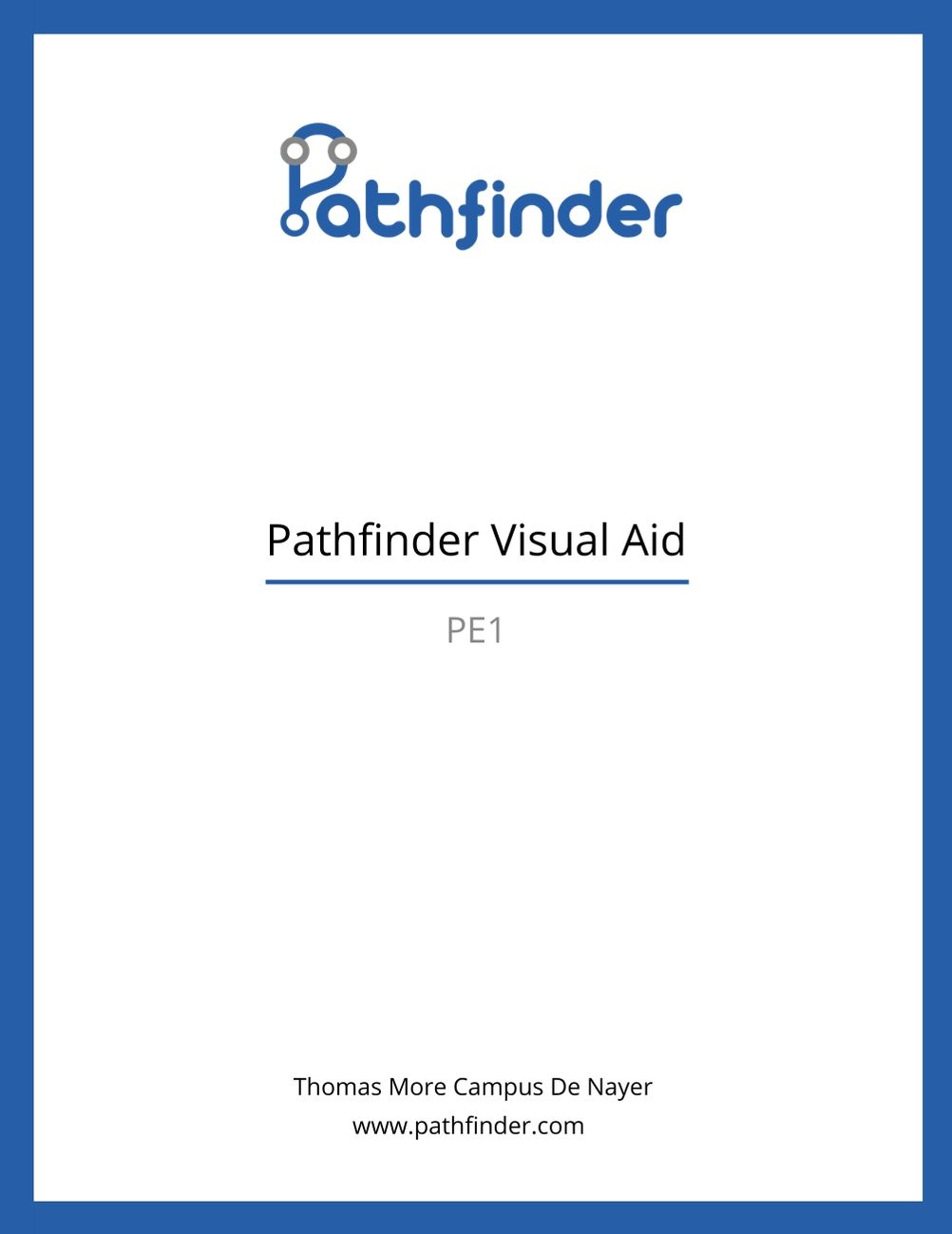
**pop psw**

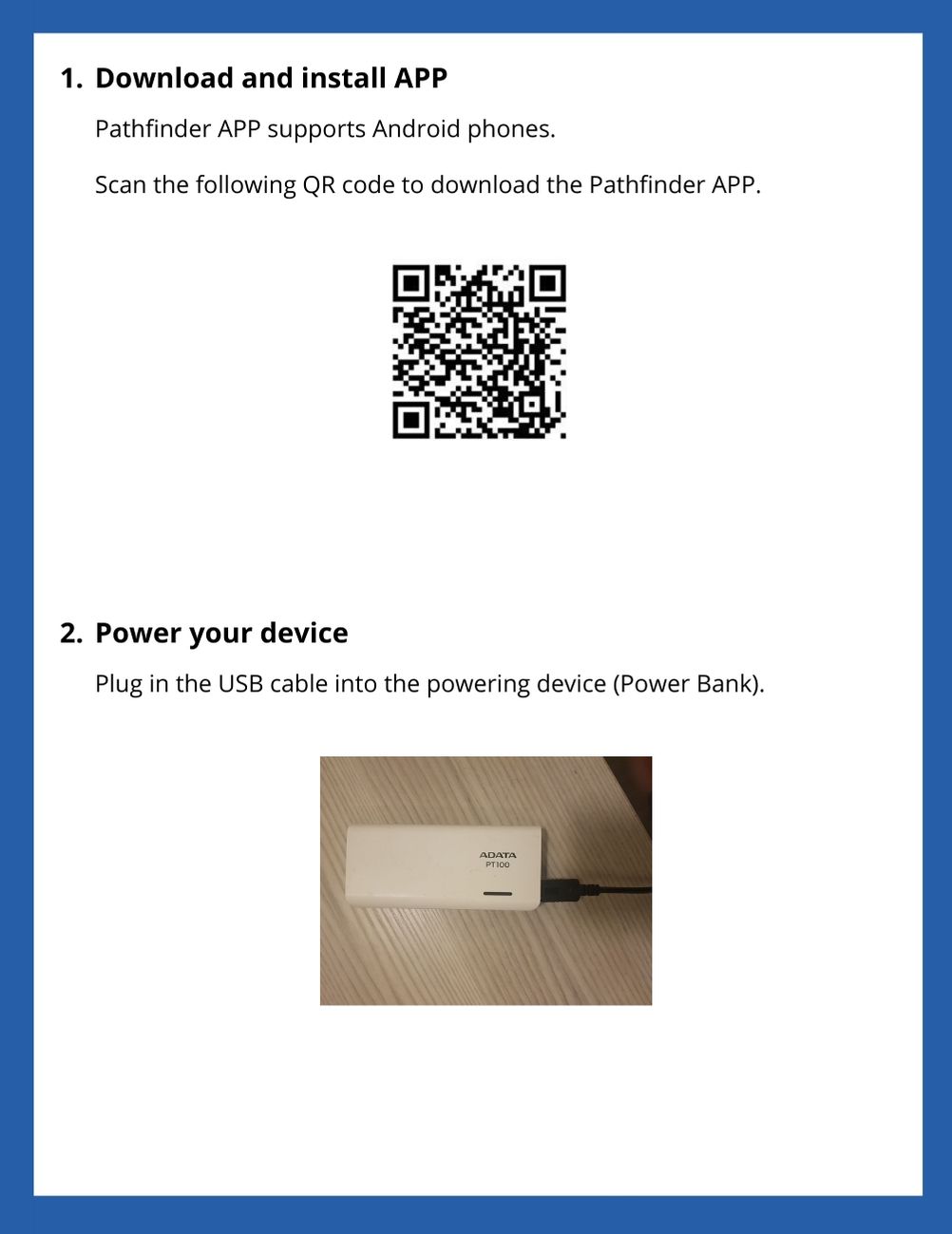
**pop acc**

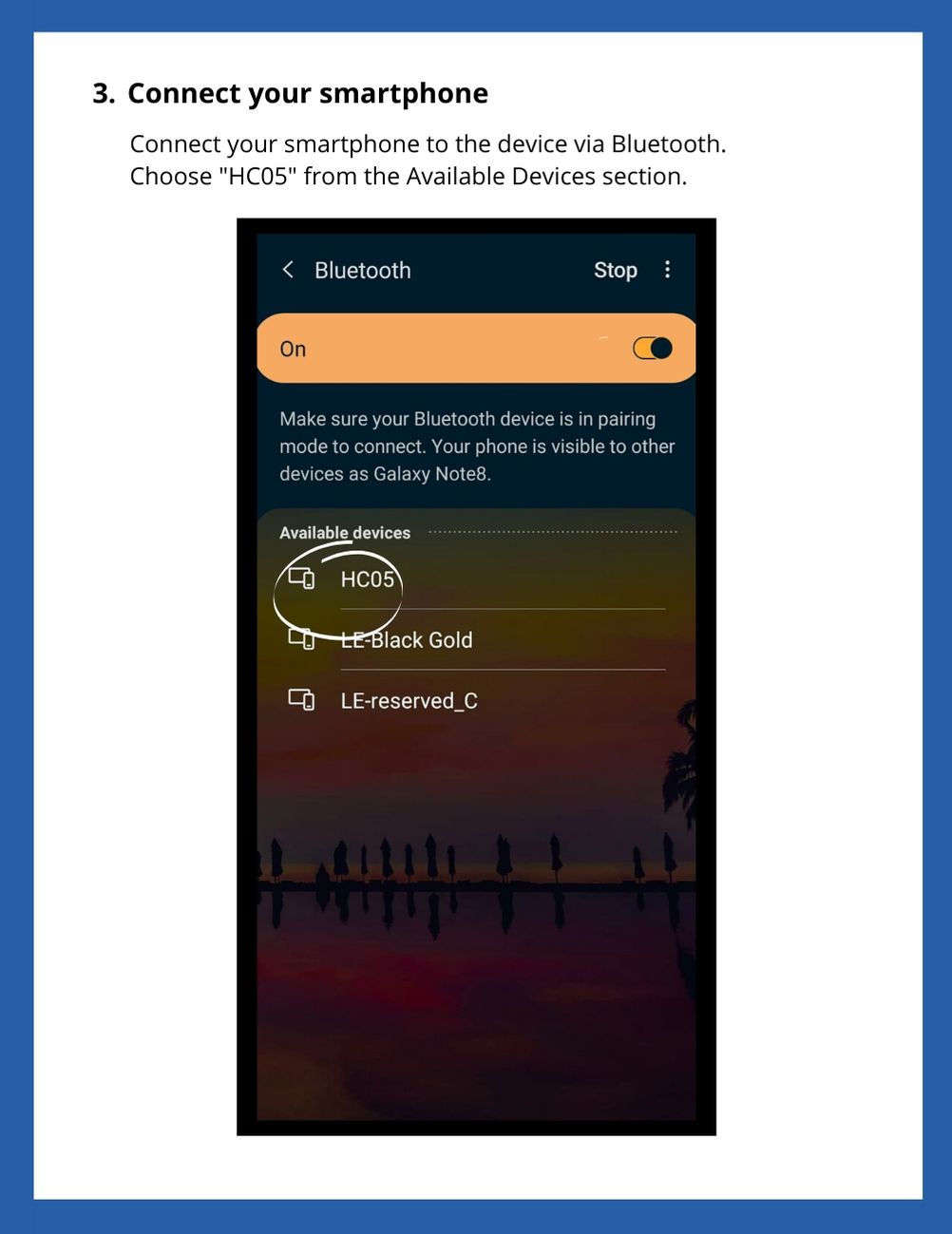
**reti ;end interrupt routine**

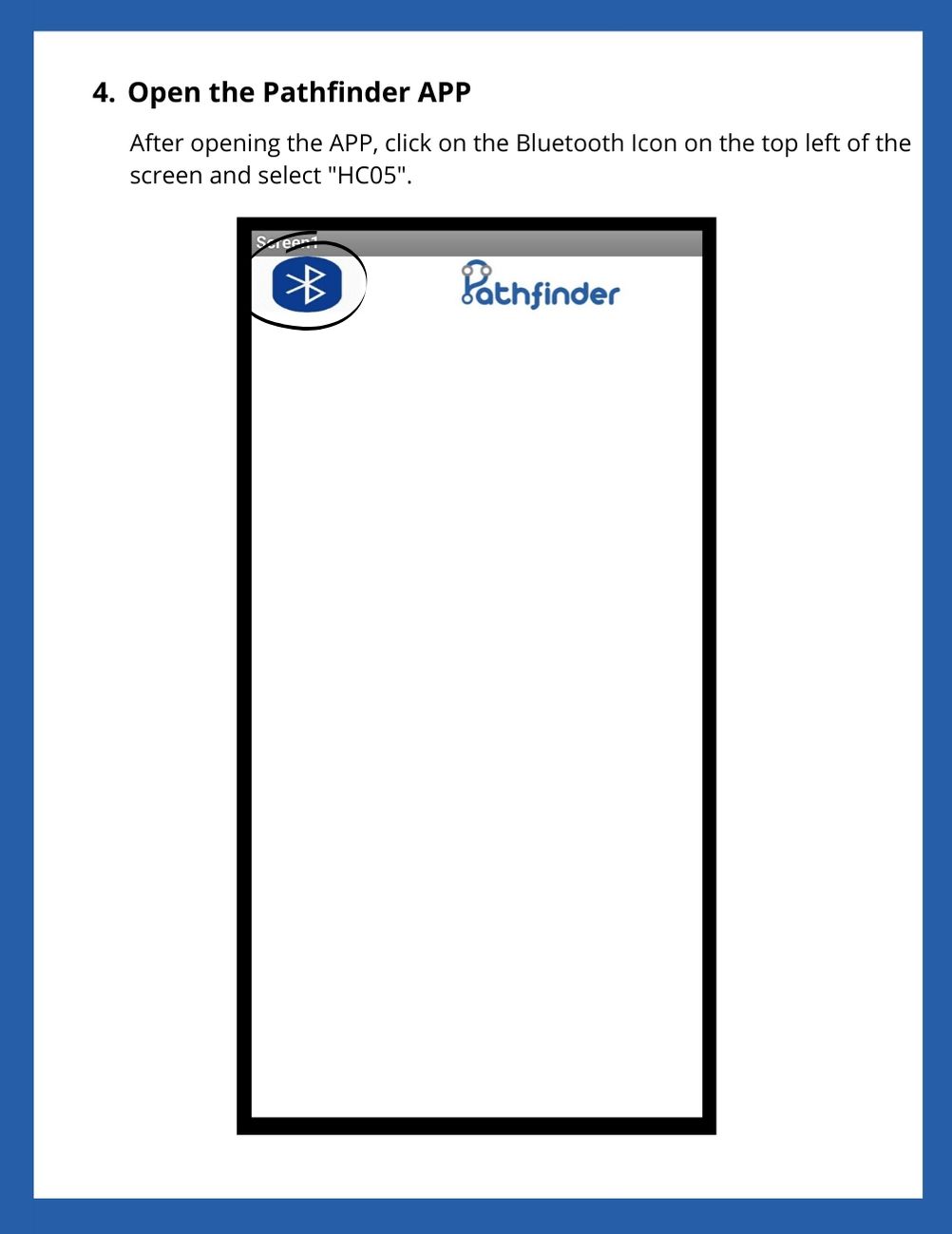
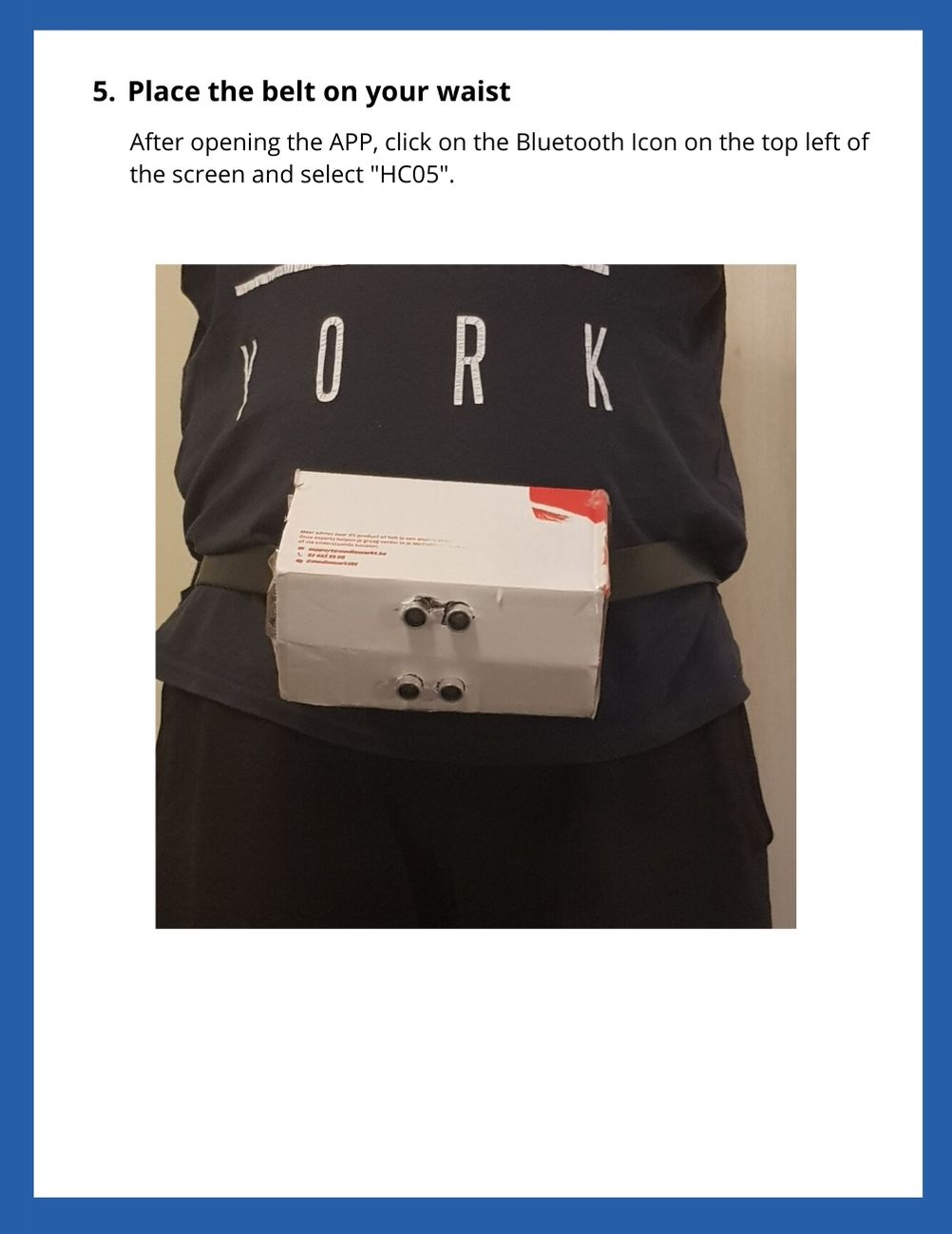
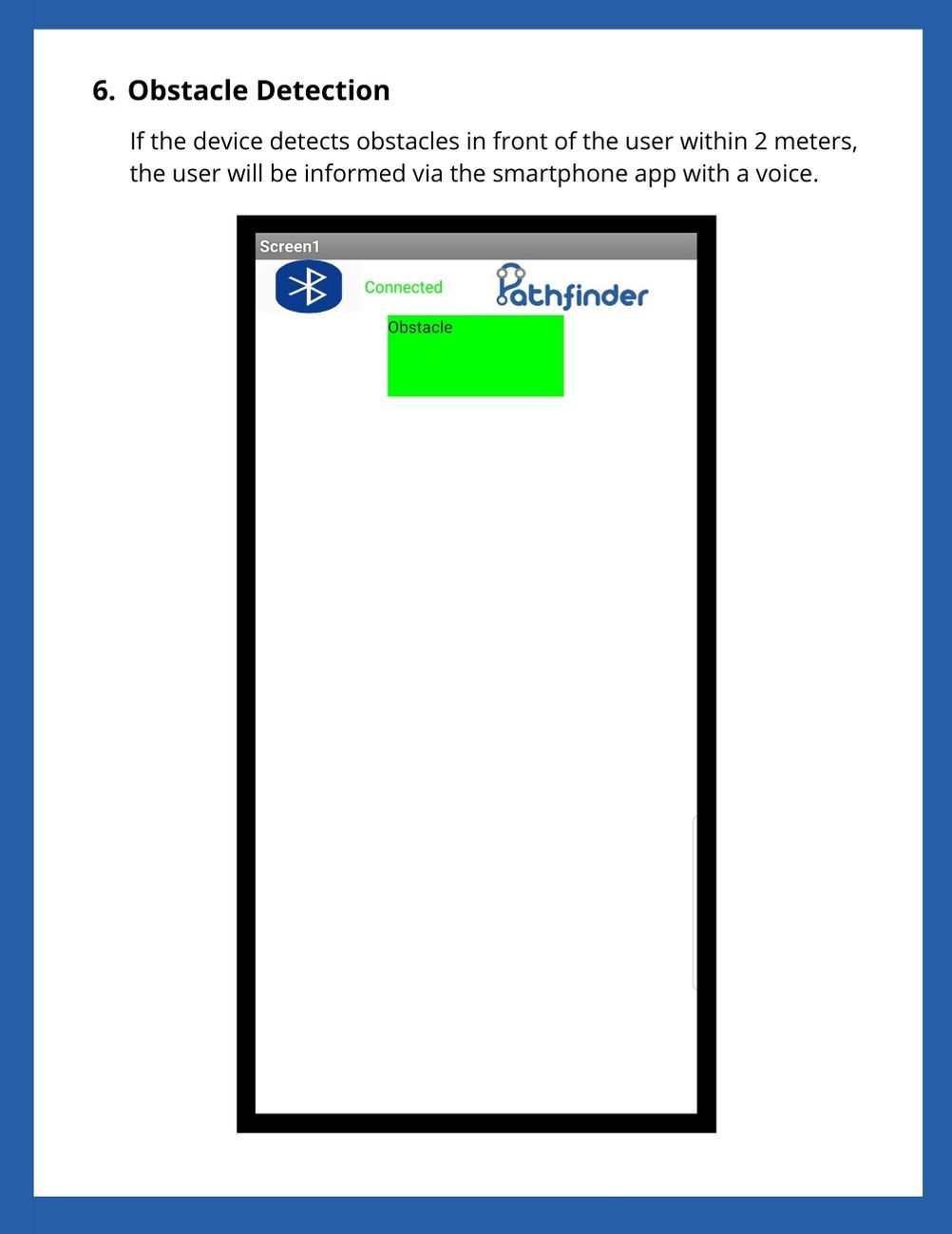
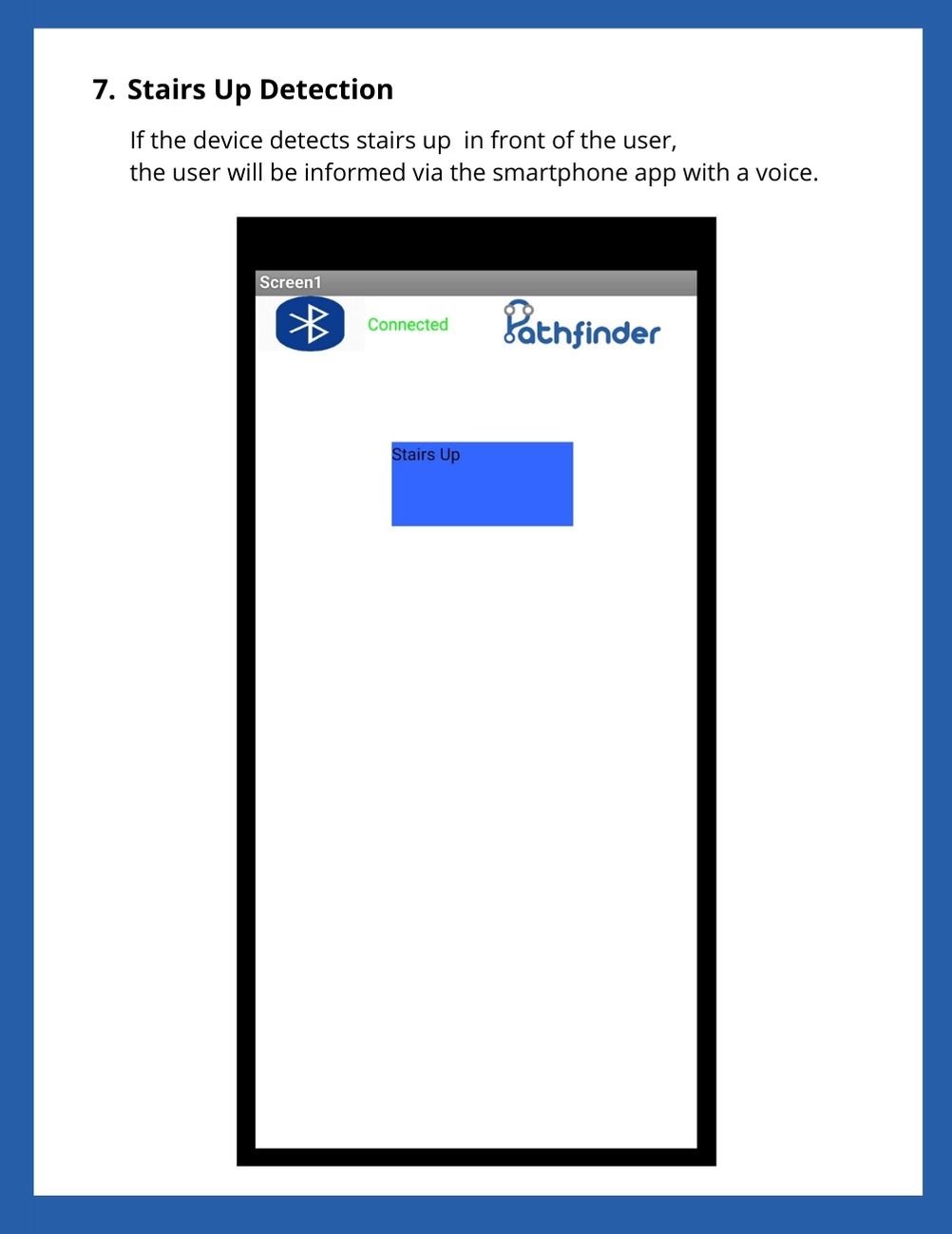
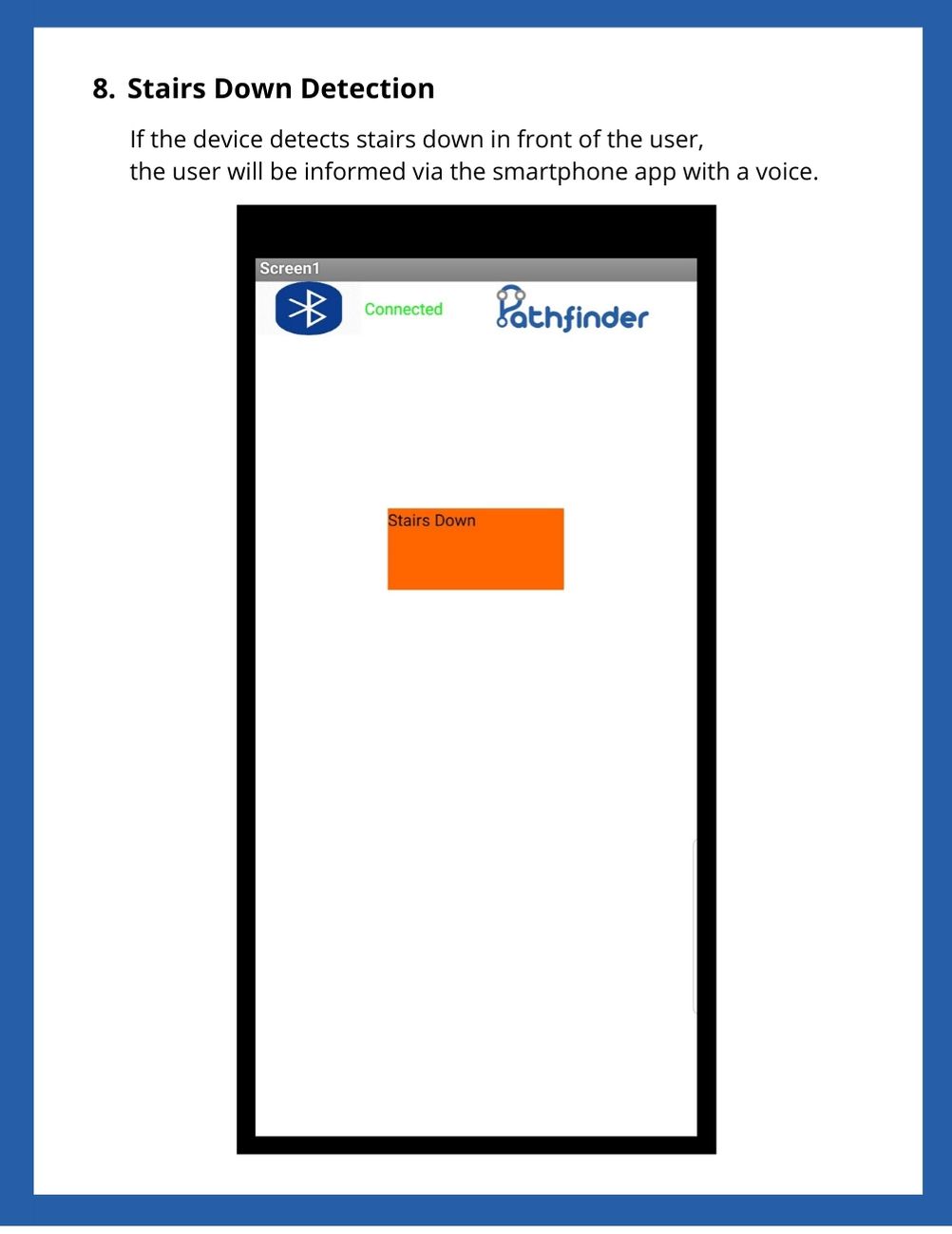
**#include "c:\xcez4.inc"**

**User manual:**

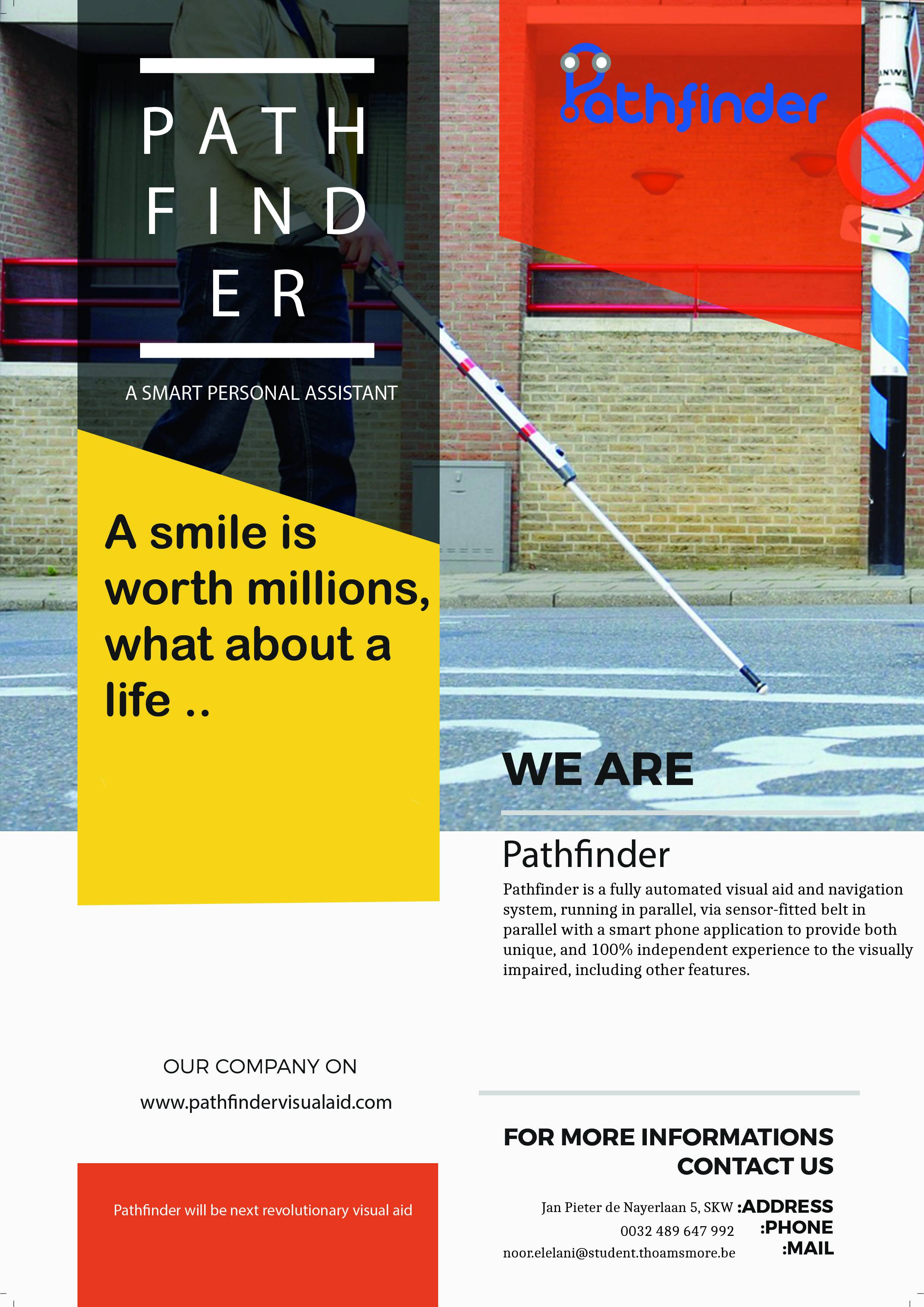






**Poster:**



**Decision:**

**Was the target met?**Yes and no, it’s functional most of the time but the ultrasonic sensors are so volatile, the solution to solve this problem is to use lidar sensors or object detection(which is way out my knowledge), and the lidar sensors are out of my budget, so as a proof of concept/prototype, I would say yes, the target was met.

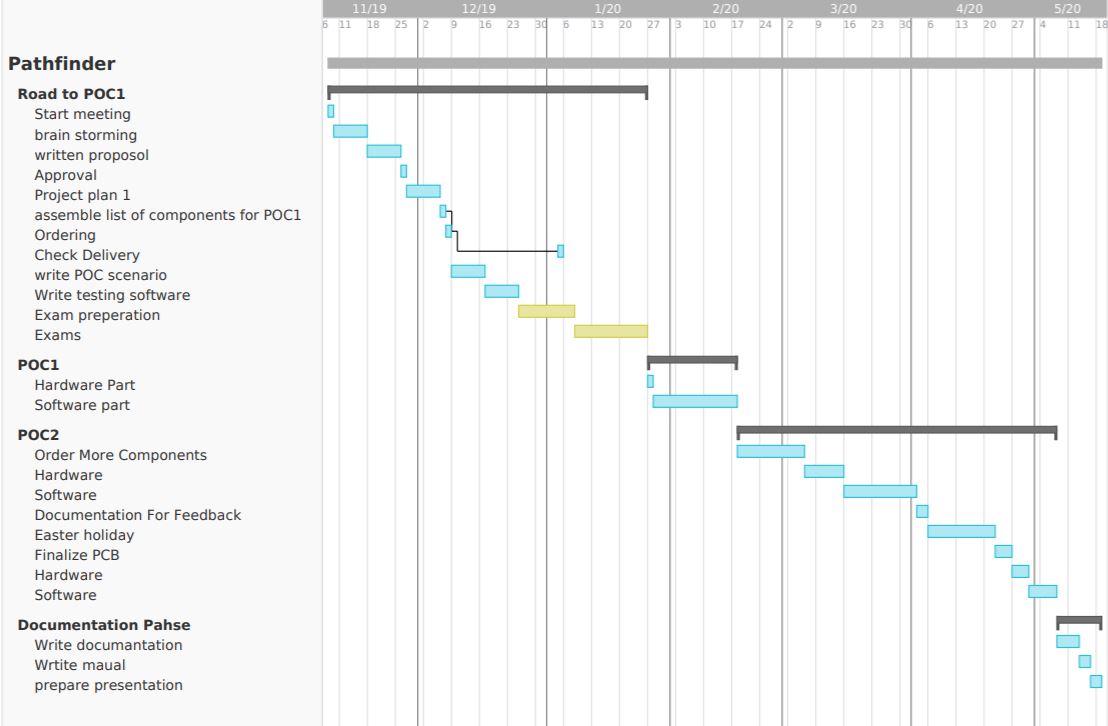
**What would you do differently next time?**

* Order components earlier
* Have a plan b in case of a pandemic
* split the time spent on the project instead of spending a lot of time on it at once (time management)
* Ask for help when stuck, instead of ruining the code more.
* Don’t underestimate a project
* Don’t underestimate the amount of time needed
* Hardware doesn’t forgive mistakes like software
* Don’t work 16 hours straight because after 8 hours straight work, you’ll end up losing productivity and focus, and that will occur in more problems.

**Reflection over the total cost: Compare with original budget.**Shipping costs got a bit out of hand, because we couldn’t split the shipping costs, but other than that it was close enough.

**Reflection to market: Where is your approach different/better/worse?**This solution is better than what’s available on the market because it uses technology which far better and more accurate than the old outdated canes, of course the device currently is just a prototype, but suppose it’s a finalized ready to go to market device I can 100% say a lot better.

**Gantt Chart:**



**Github:**

https://github.com/noorilpsp/pathfinder.git